Divide And Conquer Summary

Takeaways from D&C:

Recursive thinking can let us solve problems faster!

When solving a recursive problem, state precisely what the recursive call is giving back to you.

Use the values of the recursive call (or at least the fact you've made recursive calls) when designing the combine step

If your "combine" step isn't faster than baseline, your whole algorithm isn't better than the baseline!

Master Theorem

Given a recurrence of the following form, where a, b, c, and d are constants:

$$T(n) = \begin{cases} d & \text{if } n \text{ is at most some constant} \\ aT\left(\frac{n}{b}\right) + f(n) & \text{otherwise} \end{cases}$$

Where $f(n)$ is $\Theta(n^c \log^k n)$ for $k \geq 0$, $a \in \mathbb{Z}^+$, $c > 1$

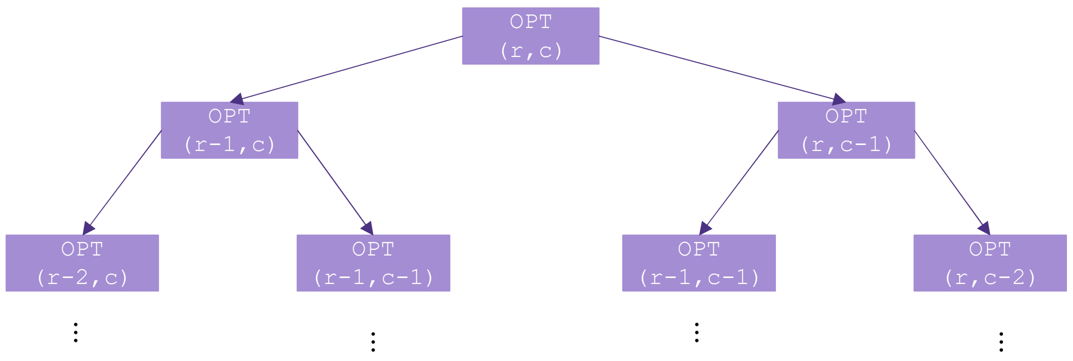
If $\log_b a < c$ then $T(n) \in \Theta(n^c \cdot \log^k n)$

If $\log_b a = c$ then $T(n) \in \Theta(n^c \log^{k+1} n)$

If $\log_b a > c$ then $T(n) \in \Theta(n^{\log_b a})$

Theorem still holds even if there are ceilings/floors in the $T\left(\frac{n}{b}\right)$ term.

Tree Method, Maybe…



When do we hit the base case?
Sometime between $\min(r,c)$ and $r+c$ levels.

Activity

Fill out the question at pollev.com/robbie

Figure out how to take advantage of the repeated calculation.
What do you think the running time will be of your new algorithm?

```
FindOPT(int i,int j, bool[][] rocks, bool[][] eggs)
   if(i<0 || j < 0) return -∞
   if(rocks[i][j]) return −∞
   if(i==0 && j==0) return eggs[0][0]
   int left = FindOPT(i-1,j,rocks,eggs)
   int down = FindOPT(i,j-1,rocks,eggs)
   return Max(left,down) + eggs[i][j]
```